strncpy_s() and strncat_s()

Daniel Plakosh, Software Engineering Institute [vita[1]]

2005-09-27; Updated 2008-07-17 L1 / D/P, L[2]

The `strncpy()` and `strncat()` functions are a source of buffer overflow vulnerabilities. The `strncpy_s()` and `strncat_s()` functions are defined in ISO/IEC TR 24731 as drop-in replacements for `strncpy()` and `strncat()`.

Development Context

Copying and concatenating character strings

Technology Context

C, UNIX, Win32

Attacks

Attacker executes arbitrary code on machine with permissions of compromised process or changes the behavior of the program.

Risk

The `strncpy()` and `strncat()` functions are a source of buffer overflow vulnerabilities.

Description

ISO/IEC TR 24731 specifies the `strncpy_s()` and `strncat_s()` functions as close replacements for `strncpy()` and `strncat()`.

The `strncpy_s()` function copies not more than a specified number of successive characters (characters that follow a null character are not copied) from a source string to a destination character array. If no null character was copied, then the last character of the destination character array is set to a null character.

The `strncpy_s()` function returns zero to indicate success. If the input arguments are invalid, `strncpy_s()` returns a nonzero value and sets the destination string to the null string. Input validation fails if either the source or destination pointers are NULL or if the maximum size of the destination string is zero or greater than `RSIZE_MAX`. The input is also considered invalid when the specified number of characters to be copied exceeds `RSIZE_MAX`.

A `strncpy_s()` operation can actually succeed when the number of characters specified to be copied exceeds the maximum length of the destination string as long as the actual source string is shorter than the maximum length of the destination string. If the number of characters to copy is greater than or equal to the maximum size of the destination string and the source string is longer than the destination buffer, the operation will fail.

Figure 1. Sample use of strncpy_s() function

```
char src1[100] = "hello";
char src2[7] = {'g','o','o','d','b','y','e'};
char dst1[6], dst2[5], dst3[5];
int r1, r2, r3;
r1 = strncpy_s(dst1, 6, src1, 100);
```

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/268-BSI.html (Plakosh, Daniel)

```
6. r2 = strncpy_s(dst2, 5, src2, 7);
7. r3 = strncpy_s(dst3, 5, src2, 4);
```

Users of these functions are less likely to introduce a security flaw because the size of the destination buffer and the maximum number of characters to append must be specified. The `strncat_s()` function also ensures null termination of the destination string. For example, the first call to `strncpy_s()` on line 5 of the sample program shown in Figure 1 assigns the value zero to r1 and the sequence hello\0 to dst1. The second call on line 6 assigns a non-zero value to r2 and the sequence \0 to dst2. The third call on line 7 assigns the value zero to r3 and the sequence good\0 to dst3. If `strncpy()` had been used instead of `strncpy_s()`, a buffer overflow would have occurred during the execution of line 6.

The `strncat_s()` function appends not more than a specified number of successive characters (characters that follow a null character are not copied) from a source string to a destination character array. The initial character from the source string overwrites the null character at the end of the destination array. If no null character was copied from the source string, then a null character is written at the end of the appended string.

The `strncat_s()` function fails and returns a nonzero value if either the source or destination pointers are NULL or if the maximum length of the destination buffer is equal to zero or greater than `RSIZE_MAX`. The function also fails when the destination string is already full or if there is not enough room to fully append the source string.

The `strncpy_s()` and `strncat_s()` functions are still capable of overflowing a buffer if the maximum length of the destination buffer and number of characters to copy are incorrectly specified.

References

[ISO/IEC 99]	ISO/IEC. *ISO/IEC 9899 Second edition 1999-12-01 Programming languages — C*. International Organization for Standardization, 1999.
[ISO/IEC 04]	ISO/IEC. *ISO/IEC WDTR 24731 Specification for Secure C Library Functions*. International Organization for Standardization, 2004.

Pearson Education, Inc. Copyright